

Search via JSON Body

Goal

Support a new way to specify a search

```
POST -H 'Content-Type: application/json'  
/search/collections -d '{ "condition": { "and":  
[ { "not": { "or": [ { "provider": "GCMDTEST" },  
{ "and": [ {  
"project": "test-campaign",  
"platform": "mars-satellite" } ] } ] } } } ,  
{ "bounding_box": [ -45,15,0,25 ],  
"revision_date":  
[ null, "2015-04-01T00:00:000Z" ] } } } '
```

Traceability

 CMR-1600 - JIRA project doesn't exist or you don't have permission to view it.

Endpoint

- POST <http://localhost:3003/collections>
 - NOTE: We already allow POST which expects a query parameter string as a body so need to require the content-type header of 'application/json' for this route
- JSON Body - detailed section below
- Accepts some query parameters similar to AQL
 - page_size, page_num, pretty, include_facets, has_granules, include_granule_counts, hierarchical_facets (no echo_compatible)
 - Perform validation that no other query parameters are provided (other parameters must be in the body)
 - **Jason suggested moving this into the JSON body in the future**
- Initial implementation only for collections
- For response format we read Accept header or use extension identical to parameters search route

JSON Body

- Allows for AND/OR/NOT and grouping capabilities
- Supports all of the same parameters as the query parameter style
- Values can be any of the following types
 - Strings
 - Integers
 - Floats
 - Booleans
 - Null
 - Maps
 - Arrays consisting of any of the valid types
- Dates are strings
 - Date ranges are an array of strings
 - Example - any date prior to 2015-04-01

```
[null, "2015-04-01T00:00:000Z"]
```

- Alternatives
 - Provide some kind of range query like:

```
{"revision-date" : {"range" : { "gte" : "2013-01-01",  
"lt" : "2014-01-01" }}}}
```

- **TODO: Look into GeoJSON for spatial fields - see <http://geojson.org/geojson-spec.html#appendix-a-geometry-examples>**
- The following options are supported for string parameters
 - ignore_case - bool
 - pattern - bool
- When passing options the map should include a value and any options

```
{ "value": <value>,  
"pattern": <true|false> }
```

- Valid keys in a map include any of the parameters supported on the query parameter API, 'and', 'or', 'not'
- Keys within the same map are implicitly and'ed
- Parameters
 - Strings
 - Example:

```
{"entry-title": "The entry-title"}
```

- Fields
 - entry-title
 - entry-id
 - provider
 - short-name
 - version (versions can be "001" for example)
 - processing-level-id
 - concept-id
 - platform
 - instrument
 - sensor
 - project
 - archive-center
 - spatial-keyword
 - two-d-coordinate-system-name
 - dif-entry-id (string but special case to look for entry-id or associated-difs with that ID)
 - collection-data-type (string but special case for SCIENCE_QUALITY) really an enumeration
 - keyword - String search but always using wild cards and case insensitive
- Booleans
 - Example:

```
{"downloadable": true}
```

- Fields
 - downloadable
 - browsable
- attribute - **Review as maps**
 - search by attribute name

```
{"attribute": {"name": "PERCENTAGE"}}
```

- search by attribute name, type, and exact value

```
{"attribute": {"type": "float",
    "name": "PERCENTAGE",
    "value": 25.5}}
```

- search by attribute name, type, and range

```
{"attribute": {"type": "float",
    "name": "PERCENTAGE",
    "min-value": 25.5,
    "max-value": 30}}
```

- search by attribute name, type, min value

```
{"attribute": {"type": "float",
    "name": "PERCENTAGE",
    "min-value": 25.5}}
```

- search by attribute name, type, max value

```
{"attribute": {"type": "float",
    "name": "PERCENTAGE",
    "max-value": 30}}
```

- updated-since {"updated-since": "2012-01-05T05:01:00Z"}
- revision-date {"revision-date": ["2012-01-05T05:01:00Z", "2014-01-05T05:01:00Z"]}
- temporal - **TODO Change to maps (start_date, end_date, recurring_start_day, recurring_stop_day), support for ISO 8601**
 - Supports additional option exclude_boundary
 - Examples
 - {"temporal": ["2012-01-05T05:01:00Z", "2014-01-05T05:01:00Z", 30, null]}
 - {"temporal": {"value": ["2012-01-05T05:01:00Z", "2014-01-05T05:01:00Z", null, null], "exclude_boundary": true}}
- tiling-system - **Need to review**
 - ':' is the separator between name and coordinates; range is indicated by '...', otherwise it is a single value.
 - curl "http://localhost:3003/granules?grid[]>wrs-1:5,10:8-10,0-10
 - Example

```
{"tiling_system": {"name": "wrs-1",
    "coordinates": [ [5,10], [8-10,0-10] ]}}
```

- science-keywords - **Review as map**
 - Map with keys for each of the subfields
 - Examples

- { "science_keywords": { "category": "EARTH SCIENCE",
 "topic": "ATMOSPHERE",
 "term": "CLOUDS",
 "variable_level_1": "VAR-L1",
 "variable_level_2": "VAR-L2",
 "variable_level_3": "VAR-L3",
 "detailed_variable": "DETAILED-VAR" } }

```
{ "science_keywords": { "topic": "OCEANS",
    "variable_level_3": "BALEEN WHALES" } }
```

- Alternatives
 - String with | separator between fields - {"science_keywords": "Earth Science|Atmosphere|Clouds|||"}
- polygon - **TODO GeoJSON**
 - Polygon points are provided in counter-clockwise order. The last point should match the first point to close the polygon. The values are an array in longitude latitude order, i.e. lon1, lat1, lon2, lat2, lon3, lat3, and so on.
 - Example * {"polygon": [10,10,30,10,30,20,10,20,10,10]}
- bounding-box
 - Bounding boxes define an area on the earth aligned with longitude and latitude. The Bounding box parameters must be a list of 4 numbers: lower left longitude, lower left latitude, upper right longitude, upper right latitude.
 - Examples
 - { "bounding_box": [-10,-5,10,5] }

```
{ "bounding_box": { "west": -10,
    "south": -5,
    "east": 10,
    "north": 5 } }
```

- point - **TODO look into GeoJSON - potentially use a map with lat lon**
 - Search using a point involves using a pair of values representing the point coordinates as parameters. The first value is the longitude and second value is the latitude.
 - Example {"point": [100, 20]}
- line - **TODO look into GeoJSON**
 - Lines are provided as a list of values representing coordinates of points along the line. The coordinates are ordered in the format lon1, lat1, lon2, lat2, lon3, lat3, and so on.
 - Example {"line": [-0.37,-14.07,4.75,1.27,25.13,-15.51]}
- **TODO - Review sort_key**
 - Array with results sorted by first field, then second for any which are identical on the first field, then third...
 - '-' indicates descending order. '+' indicates ascending order though '+' is not required because ascending is the default
 - Example

```
{"sort_key": [-start-date, entry_title]}
```

Example queries

- provider=NSIDC_ECS AND keyword=ICE

```
{ "provider": "NSIDC_ECS",
  "keyword": "ICE" }
```

- provider != GCMDTEST, does not have a project of 'test-campaign' or platform of 'mars-satellite'. Have specified bounding box and have a revision date prior to 2015-04-01.

```
{ "and": [ { "not": { "or": [ { "provider": "GCMDTEST" },
    { "and": [ { "project": "test-campaign",
      "platform": "mars-satellite" } ] } ] },
  ; implicit "and"
  { "bounding_box" : [-45,15,0,25],
    "revision_date" : [null,"2015-04-01T00:00:00Z"] } ] }
```

- wildcards and case insensitive, will match any of the _ECS providers

```
{ "provider": { "value" : "*e?s",
  "ignore_case" : true,
  "pattern" : true } }
```

JSON schema

- Use JSON schema <http://json-schema.org/> to document JSON search format
- Validate queries using existing JSON schema validation libraries as much as possible

Code design

- Add new namespace similar to AQL and Parameter parsing
 - cmr.search.services.json-parameters
 - Follow the same patterns to convert to query models as with AQL and parameters
- Tests
 - Define JSON Schema and take advantage of validations it provides
 - Many unit tests verifying JSON is correctly converted to query models - should have a test for every parameter
 - Fewer integration tests

Error rendering macro 'pageapproval' : null